

# The Software Obsolescence Minefield

## A Guide to supporting software and electronic information

Do you support products designed using software based systems?

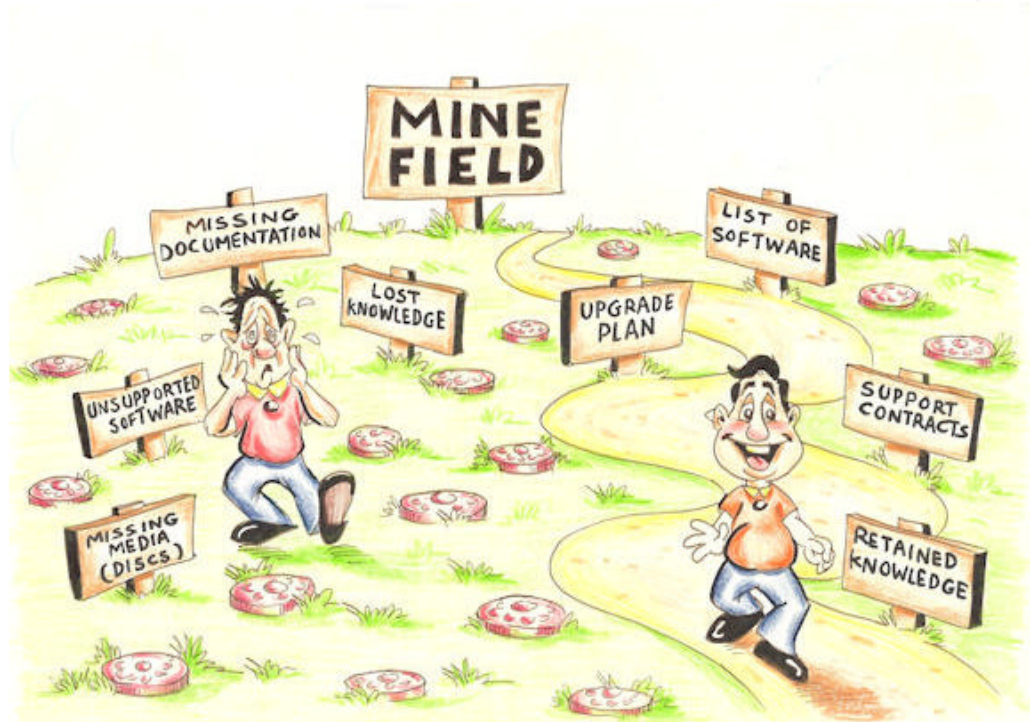
Are you managing hardware obsolescence issues for products containing software?

Do you own software which is not the latest version?

Are the software engineers from your last project no longer available?

Do other people own the IPR for the software that you use?

**If the answer is YES to any of the above questions, then you need to read this booklet.**



Published by the Component Obsolescence Group  
Unit 3, Curo Park, Frogmore, St. Albans, Herts AL2 2DD, UK  
Tel: +44 (0)1727 876029 Fax: +44 (0)1727 871336  
e-mail: [admin@cog.org.uk](mailto:admin@cog.org.uk) Web: [www.cog.org.uk](http://www.cog.org.uk)



**Price £10 €15**

## Contents

Introduction .....	2
What is software? .....	4
The starting point.....	4
Process overview .....	5
Identify your software.....	5
Assess the software obsolescence risks .....	7
Media management .....	10
Disaster recovery .....	12
Digital preservation.....	12
Software replication .....	17
Software testing .....	18
Legacy languages .....	19
Reverse engineering and code conversion.....	20
Use of support (software maintenance) contracts .....	21
Conclusions .....	23
Acronyms, terms and definitions.....	23
References and further reading .....	24
Appendix 1 – Example software obsolescence management plan – Software identification sheet.....	25
Appendix 2 – Example software obsolescence management plan - Strategy selection for software obsolescence management.....	25

The publication is one of a series of booklets published by the Component Obsolescence Group, all of which are recommended as essential reading for organisations or individuals tasked with obsolescence management. This series includes:

**The Obsolescence Minefield**  
**The Date Coding Minefield**  
**The Supply Chain Minefield**  
**The Long-Term Storage Minefield**  
**The Pb-Free Minefield**  
**The Redundant Stock Minefield**  
**The Hardware Design Minefield**

Contact the Component Obsolescence Group for details of the latest available titles

This booklet was written by **Graeme Rumney**, Sellafeld Ltd in co-operation with members of the Software Group of the Component Obsolescence Group (COG). The inputs from Raz Butt, Ben Gordon, Alan Gowland and Lloyd Francis are particularly appreciated.

Original Illustrations by Steve Padgham

© 2007 **Component Obsolescence Group International Ltd.**

## Introduction

**Software impacts on almost every aspect of our lives, from MP3 players or personal computers through to microwaves and washing machines. In manufacturing, software based systems provide control of production plants and processes, and it is also behind the sales and financial systems. Cars use software extensively in everything from engine management systems to the heating and ventilation controls. In the aerospace industry it is software that's keeping the state of the art airliner or fighter jet in the air.**

**So, can software become obsolete, and if so how can we avoid the consequences if it does?**

Software ultimately ends up as a set of binary instructions and data, which is used by a microprocessor to perform some defined function. Before it reaches the microprocessor it is created and stored on media, such as hard disks, floppy disks, Compact Discs (CDs), tape or an Electrically Erasable Programmable Read Only Memory (EEPROM). In the conventional sense, software cannot go obsolete in the same way that a component goes out of production, as software can be simply copied. However, in reality, the obsolescence mechanism for software is much more subtle and has the potential to leave your application unsupported, and could stop your business or project or programme in its tracks. During the development of software applications there is an underlying infrastructure in place which includes:

1. source code and configuration information;
2. requirements;
3. specifications;
4. design and other documentation;

5. data;
6. compilers;
7. operating systems;
8. hardware platforms and dongles;
9. storage media;
10. third party vendors; and,
11. Engineers skilled in the particular programming language.

At the end of the development of the software application, any one of these entities may not be retained by the responsible organisation and as soon as this happens, the software has already begun to develop potential obsolescence. Simply storing media with the software on it is not enough! Consideration should be given to retaining the entire underlying infrastructure, should the software require modification and re-implementation at any given point in the future. For example, try opening a user requirements specification which was written fifteen years ago on a Disk Operating System (DOS) based word processing application with your current Personal Computer (PC) – the chances are that even if it does open, it will be almost unintelligible because of the embedded formatting characters associated with the now obsolete word processing application. It could be argued that keeping a copy of the word processor application as well as the document would mitigate this risk, but will the word processor application run on a modern operating system? Probably not! Take this a stage further to the custom software which has been developed for an application, which now requires a modification after a bug has been revealed. Does the necessary infrastructure to modify the software and, importantly, engineers competent in the obsolete software language and processes still exist?

The Software Obsolescence Minefield  
Published by the Component Obsolescence Group  
Unit 3, Curo Park, Frogmore, St. Albans, Herts AL2 2DD, UK  
Tel: +44 (0)1727 876029 Fax: +44 (0)1727 871336  
e-mail: [admin@cog.org.uk](mailto:admin@cog.org.uk) Web: [www.cog.org.uk](http://www.cog.org.uk)

